IJMTST | Volume: 2 | Issue: 04 | April 2016 | ISSN: 2455-3778

Generalized Parallel CRC Computation

D.Venkanna Babu[1] | M.CH.P.V.L.Kumar[2] | M.M.Venkatesh[3] | B.Surya Prakash[4]

[1]Assistant Professor, Department of Electronics and Communication Engineering, Ramachandra College of Engineering, Eluru, Andhra Pradesh, India.
[2,3,4]B.Tech, Students, Department of Electronics and Communication Engineering, Ramachandra College of Engineering, Eluru, Andhra Pradesh, India.

ABSTRACT

This paper presents a theoretical result in the context of realizing high-speed hardware for parallel CRC checksums. Starting from the serial implementation widely reported in the literature, we have identified a recursive formula from the degree of the polynomial generator. Last, we from which our parallel implementation is derived. In comparison with previous works, the new scheme is faster and more compact and is independent of the technology used in its realization. In our solution, the number of bits processed in parallel can be different have also developed high-level parametric codes that are capable of generating the circuits autonomously when only the polynomial is given.

KEYWORDS: *Parallel CRC, LFSR, error-detection, VLSI, FPGA, V HDL, digital logic.*

I. INTRODUCTION

Cyclic Redundancy Check (CRC) is widely used in data communications and storage devices as a powerful method for dealing with data errors. It is also applied to many other fields such as the testing of integrated circuits and the detection of logical faults [6]. One of the more established hardware solutions for CRC calculation is the Linear Feedback Shift Register (LFSR), consisting of a few flip-flops (FFs) and logic gates. This simple architecture processes bits serially. In some situations, such as high-speed data communications, the speed of this serial implementation is absolutely inadequate. In these cases, a parallel computation of the CRC, where successive units of w bits are handled simultaneously, is necessary or desirable. Like any other combinatorial circuit, parallel CRC hardware could be synthesized with only two levels of gates. This is defined by laws governing digital logic. Unfortunately, this implies a huge number of gates. Furthermore, the minimization of the number of gates is an NP-hard optimization problem. Therefore, when complex circuits must be realized, one generally uses heuristics or seeks customized solutions. This paper presents a customized, elegant, and concise formal solution for building parallel CRC hardware. The new scheme generalizes and improves previous works. By making use of some mathematical principles, we will derive a recursive formula that can be used to deduce the parallel CRC circuits. Furthermore, we will show how to apply this formula and to generate the CRC circuits automatically. As in modern synthesis tools, where it is possible to specify the number of inputs of an adder and automatically generate necessary logic, we developed the necessary parametric codes to perform the same tasks with parallel CRC circuits. The compact representation proposed in the new scheme provides the possibility of significantly saving hardware and reaching higher frequencies in comparison with previous works. Finally, in our solution, the degree of the polynomial generator, m, and the number of bits processed in parallel, w, can be different. The article is structured as follows: Section 2 illustrates the key elements of CRC. In Section 3, we summarize previous works on parallel CRCs to provide appropriate background. In Section 4, we derive our logic equations and present the parallel circuit. In

addition, we illustrate the performance by some examples. Finally, in Section 5, we evaluate our results by comparing them with those presented in previous works. The codes implemented are included in the Appendix.

II. Cyclic Redundancy Check

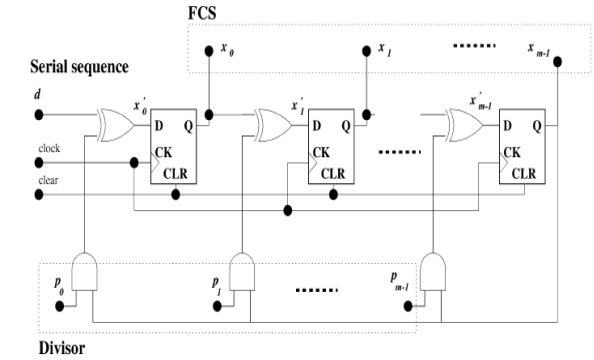
As already stated in the introduction, CRC is one of the most powerful error-detecting codes. Briefly speaking CRC can be described as follows: Let us suppose that a transmitter, T, sends a sequence, S1, of k bits,fb0;b1;;bkÿ1g, to a receiver, R. At the same time, T generates another sequence, S2, of m bits, fb0 0;b 0 1;;b 0 mÿ1g, to allow the receiver to detect possible errors. The sequence S2 is commonly known as a Frame Check Sequence (FCS). It is generated by taking into account the fact that the complete sequence, S ¼ S1 [S2, obtained by the concatenating of S1 and S2, has the property that it is divisible (following a particular arithmetic) by some predetermined sequence P, fp0;p1;;, ofmÞ1 bits. After T sends S to R. R divides S (i.e., the message and the FCS)byP,usingthesameparticulararithmetic.Afterit receivesthe message. If there is no remainder, R assumes there was no error. Fig. 1 illustrates how this mechanism works. A modulo 2 arithmetic is used in the digital realization of the above concepts [3]: The product operator is accomplished by a bitwise AND, whereas both the sum and subtraction are accomplished by bitwise XOR operators. In this case, a CRC circuit (modulo 2 another possible implementation of the CRC circuit [7] is shown in Fig. 3. In this paper, we will call it LFSR2. In this circuit, the outputs of FFs (after k clock periods) are the same FCS computed by LFSR. It should be mentioned that, when LFSR2 is used, no sequence of m zeros has to be sent through d. So, LFSR2 computes FCS faster than LFSR. In practice, the message length is usually much greater than m, so LFSR2 and LFSR have similar performancedivisor) can be easily realized as a special shift register, called LFSR. Fig. 2 shows a typical architecture. It can be used by both the transmitter and receiver.

In Fig. 2, we show that m FFs have common clock and clear signals. The input x0 i of the ith FF is obtained by taking an XOR of the ðiÿ1Þth FF output and a term given by the logical AND between pi and xmÿ1. The signal x0 0 is obtained by taking an XOR of the input d and xmÿ1. Ifpi is zero, only a shift operation is performed (i.e., XOR related to x0 i is not required); otherwise, the feedback xmÿ1 is XOR-ed with xiÿ1. We point out that the AND gates in Fig. 2 are unnecessary if the divisor P is time-invariant. The sequence S1 is sent serially to the input d of the circuit starting from the most significant bit, b0. Let us suppose that the k bits of the sequence S1 are an integral multiple of m, the degree of the divisor P. The process begins by clearing all FFs. Then, all k bits are sent, once per clock cycle. Finally, m zero bits are sent through d. In the end, the FCS appears at the output end of the FFs.

III. Previous Methods

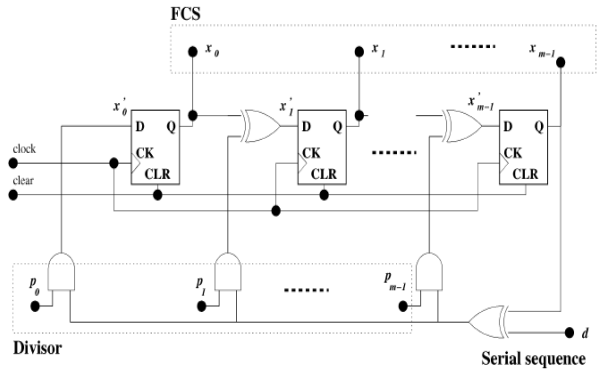
Basically, the hardware implementation for the CRC in series computation is LFSR (Linear Feedback Shift Register). It consists of FFs and logic gates

Linear Feedback Shift Register-1

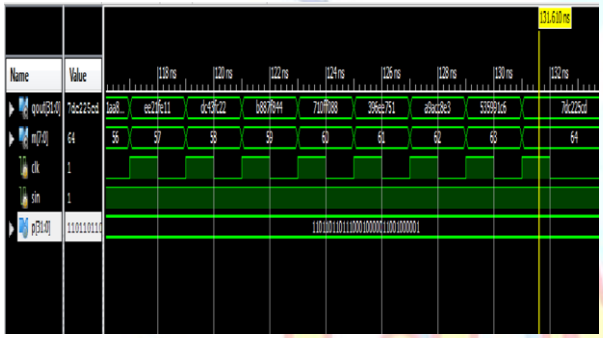


Simulation and synthesis result of LFSR-1

Linear Feedback Shift Register-2



Simulation and synthesis result of LFSR-2



Drawbacks of series CRC computation:

Implementation of series computation is inadequate. It takes more number of clock cycles to detect the errors.

IV. PROPOSED METHOD

PARALLEL CRC COMPUTATION

There are different techniques for parallel CRC generation given as follow. 1. A Table-Based Algorithm for Pipelined CRC Calculation. 2. Fast CRC Update 3. F matrix based parallel CRC generation. 4. Unfolding, Retiming and pipelining Algorithm

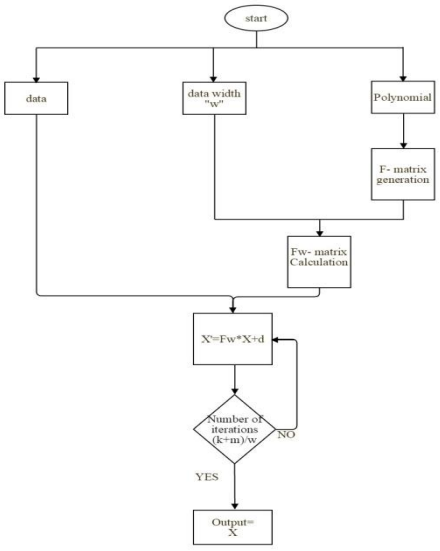
Parallel CRC is mainly depends on the F- matrix generation. Basic f-matrix for the given error polynomial

i.e.,"1101101101110001000011001000001".

Derivation of the formula for matrix F^W

$$F^i = [F^{i-1}[p_{m-1} \ldots . p_2 p_1 p_0]| \text{the first m} - 1 \text{ coloumns of } F^{i-1}]$$

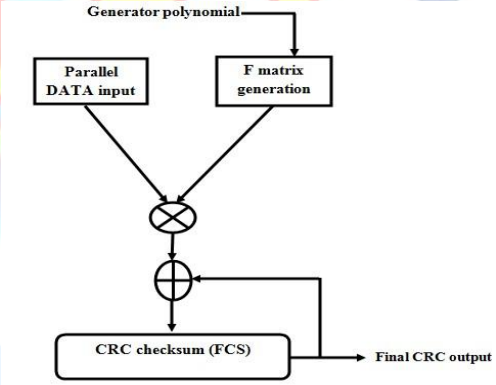
Algorithm for F- matrix:



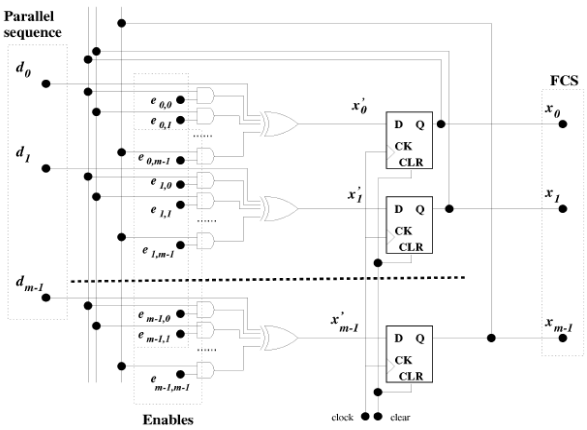
If w<m then the inputs $[d_{m-1} \ldots . d_w]$ are not needed. Inputs$[d_{w-1} \ldots . d_0]$ are the bits of dividend sent in groups of 'w' bits each.

. Where As realization of the LFSR2 it has circuit very similar to the above inputs dare XORed with the FFs outputs and results are feedbacks

Flowchart for F-matrix:

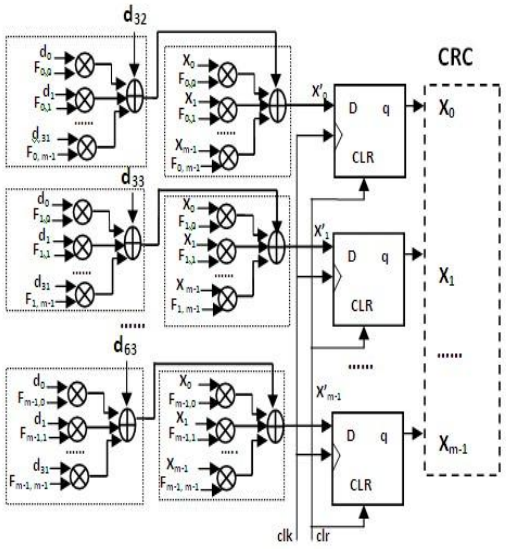


CRC -32 hardware for 32 bits processing.

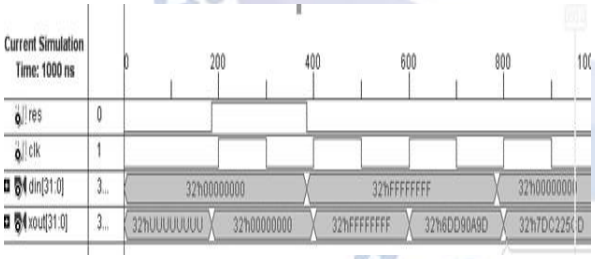


Simulation results of the CRC-32 for 32 bit parallel processing:

Block diagram of 64-bit parallel calculation of CRC-32



Simulation results of the CRC-32 for 64 bit parallel processing:



Comparison of all CRC computational models:

Type of CRC computation	Number of clock cycles used	Time period in nano seconds
LFSR-1	96	196.01ns
LFSR-2	64	131.61ns
32-Bit processing using 32-bit circuit	17	31.10ns
64-Bit processing using 32-bit circuit	9	27.42ns

V. Conclusion and Future Scope

Although this project primarily deals with the time (number of clock cycles to generate the FCS) considerations

So for high speed applications 64 bit process of CRC-32 is most required. It takes less time to generate the CRC for error detection. Here area can be reduced by replacing AND gates with NAND gates and XOR with XNOR gates. By replacing AND gate with NAND gate 2 CMOS transistor are less required. So it is the area efficient architecture. CRC-32 used in Ethernet frame for error detectionand CRC-CCITT which is 32 bit polynomial used in X-25 protocol, disc storage, SDLC, and XMODEM

References

[1] Hitesh H. Mathukiya and Naresh M. Patel “anovel approach for high speed application”- IEEE communication system and network technologies-2012.

[2] Albertengo, G.; Sisto, R.;, "Parallel CRC generation," *Micro, IEEE* , vol.10, no.5, pp.63-71,Oct1990.

[3] Braun et al. "parallel CRC computation in FPGAs," proc workshop Field Programmable Logic and Applications, 1996.